Retail Business Insights Using SQL

Uncovering Sales, Delivery, Inventory, and Customer Trends with SQL

ANKIT RAWAT

ankitrawat.ds@gmail.com

https://www.linkedin.com/in/ankitrawat-ds/

https://github.com/rawat-ankit

Project Objective

- To analyse a multi-table retail dataset using SQL
- Generate insights to improve:
  - 🧍 **Customer Retention**
  - 📦 **Inventory Planning**
  - 📈 **Sales Optimization**
  - 🚚 **Delivery Performance**
  - ⭐ **Customer Satisfaction**

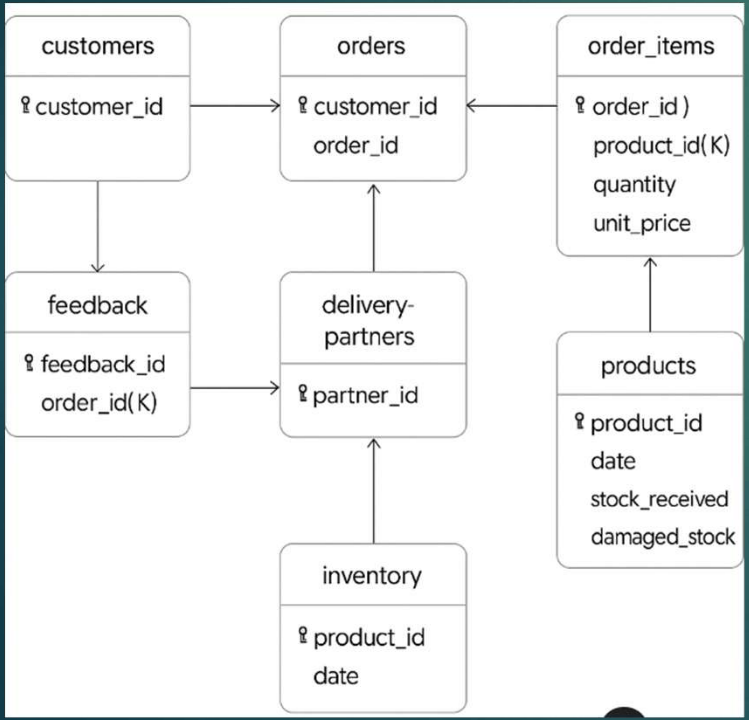
Tools Used:

- PostgreSQL
- pgAdmin

Dataset Overview & ER Diagram

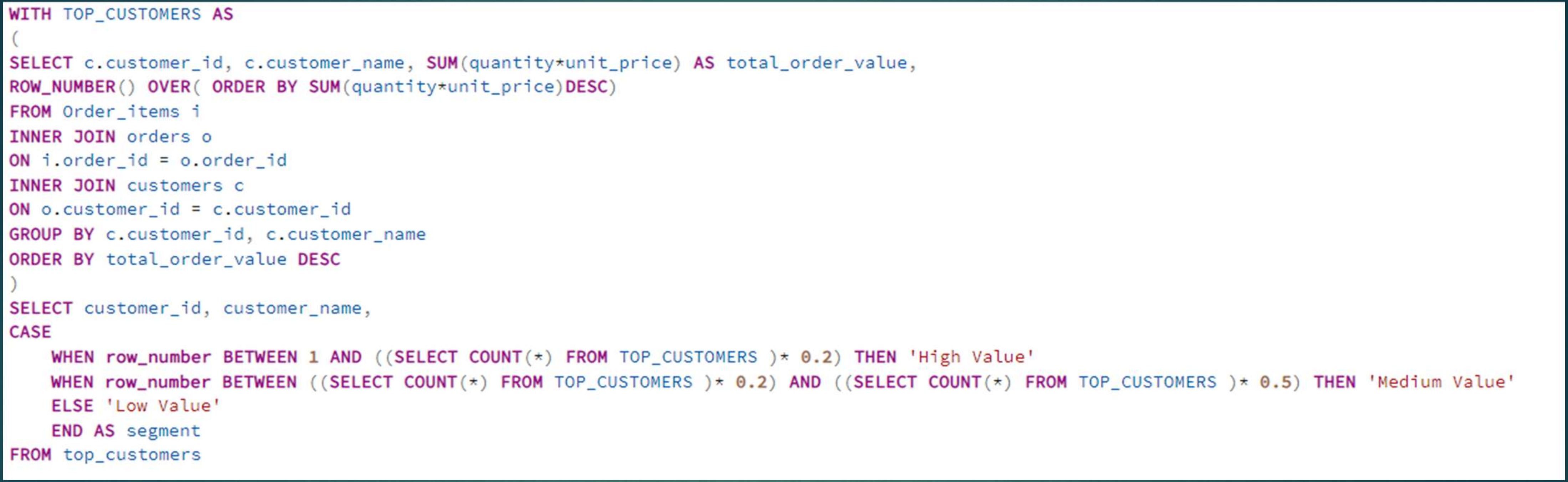
Tables Used:

- Customer, Orders, Order_items
- Products, Inventory
- Delivery_performance, Feedback

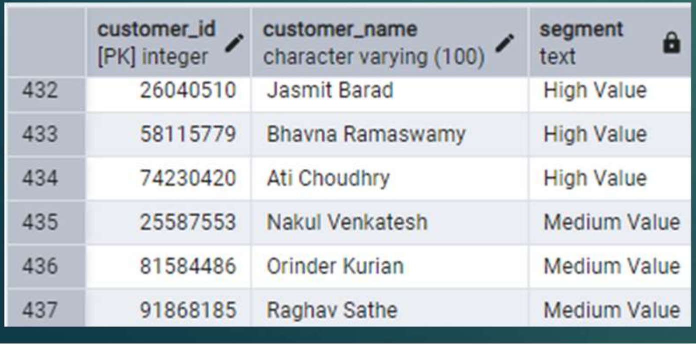


Customer Segmentation

Categorize customers based on their total spending



```
WITH TOP_CUSTOMERS AS
(
SELECT c.customer_id, c.customer_name, SUM(quantity*unit_price) AS total_order_value,
ROW_NUMBER() OVER( ORDER BY SUM(quantity*unit_price)DESC)
FROM Order_items i
INNER JOIN orders o
ON i.order_id = o.order_id
INNER JOIN customers c
ON o.customer_id = c.customer_id
GROUP BY c.customer_id, c.customer_name
ORDER BY total_order_value DESC
)
SELECT customer_id, customer_name,
CASE
    WHEN row_number BETWEEN 1 AND ((SELECT COUNT(*) FROM TOP_CUSTOMERS )* 0.2) THEN 'High Value'
    WHEN row_number BETWEEN ((SELECT COUNT(*) FROM TOP_CUSTOMERS )* 0.2) AND ((SELECT COUNT(*) FROM TOP_CUSTOMERS )* 0.5) THEN 'Medium Value'
    ELSE 'Low Value'
    END AS segment
FROM top_customers
```

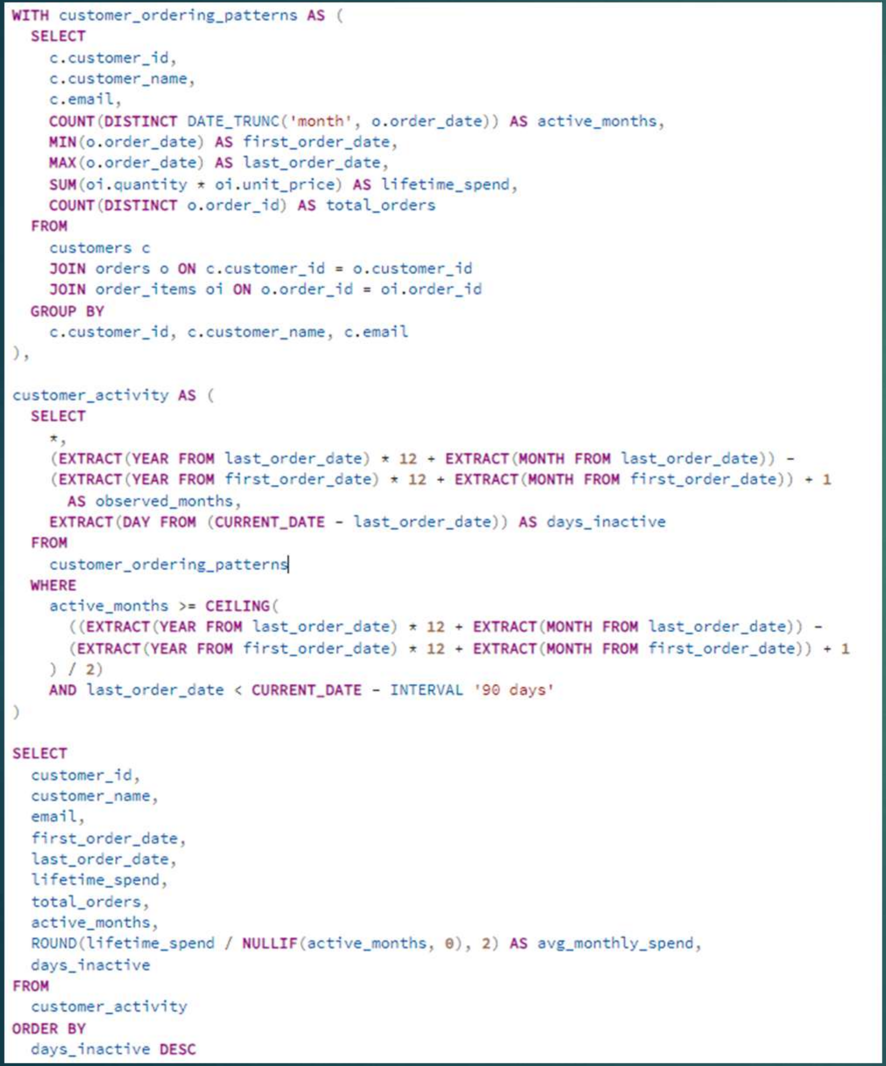


	customer_id [PK] integer	customer_name character varying (100)	segment text
432	26040510	Jasmit Barad	High Value
433	58115779	Bhavna Ramaswamy	High Value
434	74230420	Ati Choudhry	High Value
435	25587553	Nakul Venkatesh	Medium Value
436	81584486	Orinder Kurian	Medium Value
437	91868185	Raghav Sathe	Medium Value

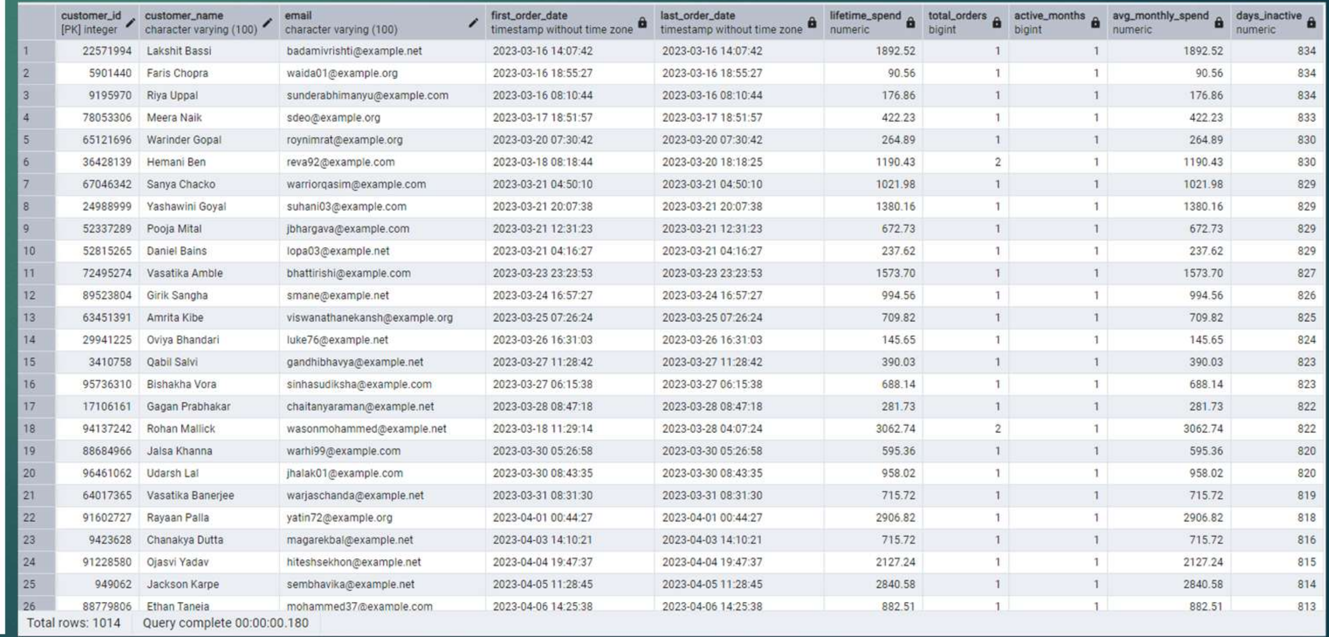
Segment into:
Top 20% → High Value
Next 30% → Medium Value
Rest → Low Value

Churn Detection

Find customers inactive for 90+ days



```
WITH customer_ordering_patterns AS (
  SELECT
    c.customer_id,
    c.customer_name,
    c.email,
    COUNT(DISTINCT DATE_TRUNC('month', o.order_date)) AS active_months,
    MIN(o.order_date) AS first_order_date,
    MAX(o.order_date) AS last_order_date,
    SUM(oi.quantity * oi.unit_price) AS lifetime_spend,
    COUNT(DISTINCT o.order_id) AS total_orders
  FROM
    customers c
    JOIN orders o ON c.customer_id = o.customer_id
    JOIN order_items oi ON o.order_id = oi.order_id
  GROUP BY
    c.customer_id, c.customer_name, c.email
),

customer_activity AS (
  SELECT
    *,
    (EXTRACT(YEAR FROM last_order_date) * 12 + EXTRACT(MONTH FROM last_order_date)) -
    (EXTRACT(YEAR FROM first_order_date) * 12 + EXTRACT(MONTH FROM first_order_date)) + 1
      AS observed_months,
    EXTRACT(DAY FROM (CURRENT_DATE - last_order_date)) AS days_inactive
  FROM
    customer_ordering_patterns
  WHERE
    active_months >= CEILING(
      ((EXTRACT(YEAR FROM last_order_date) * 12 + EXTRACT(MONTH FROM last_order_date)) -
      (EXTRACT(YEAR FROM first_order_date) * 12 + EXTRACT(MONTH FROM first_order_date)) + 1
    ) / 2)
    AND last_order_date < CURRENT_DATE - INTERVAL '90 days'
)

SELECT
  customer_id,
  customer_name,
  email,
  first_order_date,
  last_order_date,
  lifetime_spend,
  total_orders,
  active_months,
  ROUND(lifetime_spend / NULLIF(active_months, 0), 2) AS avg_monthly_spend,
  days_inactive
FROM
  customer_activity
ORDER BY
  days_inactive DESC
```

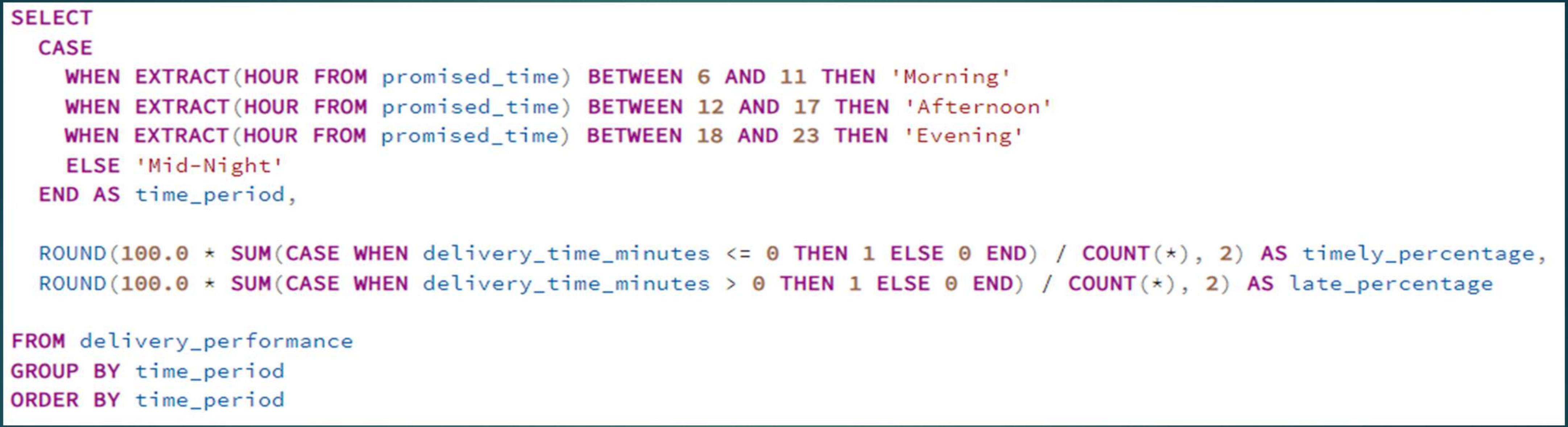


	customer_id [PK] integer	customer_name character varying (100)	email character varying (100)	first_order_date timestamp without time zone	last_order_date timestamp without time zone	lifetime_spend numeric	total_orders bigint	active_months bigint	avg_monthly_spend numeric	days_inactive numeric
1	22571994	Lakshit Bassi	badamivrishti@example.net	2023-03-16 14:07:42	2023-03-16 14:07:42	1892.52	1	1	1892.52	834
2	5901440	Faris Chopra	waida01@example.org	2023-03-16 18:55:27	2023-03-16 18:55:27	90.56	1	1	90.56	834
3	9195970	Riya Uppal	sunderabhimanyu@example.com	2023-03-16 08:10:44	2023-03-16 08:10:44	176.86	1	1	176.86	834
4	78053306	Meera Naik	sdeo@example.org	2023-03-17 18:51:57	2023-03-17 18:51:57	422.23	1	1	422.23	833
5	65121696	Warinder Gopal	roynimrat@example.org	2023-03-20 07:30:42	2023-03-20 07:30:42	264.89	1	1	264.89	830
6	36428139	Hemani Ben	reva92@example.com	2023-03-18 08:18:44	2023-03-20 18:18:25	1190.43	2	1	1190.43	830
7	67046342	Sanya Chacko	warriorqasim@example.com	2023-03-21 04:50:10	2023-03-21 04:50:10	1021.98	1	1	1021.98	829
8	24988999	Yashawini Goyal	suhani03@example.com	2023-03-21 20:07:38	2023-03-21 20:07:38	1380.16	1	1	1380.16	829
9	52337289	Pooja Mital	jbhargava@example.com	2023-03-21 12:31:23	2023-03-21 12:31:23	672.73	1	1	672.73	829
10	52815265	Daniel Bains	lopa03@example.net	2023-03-21 04:16:27	2023-03-21 04:16:27	237.62	1	1	237.62	829
11	72495274	Vasatika Amble	bhattirishi@example.com	2023-03-23 23:23:53	2023-03-23 23:23:53	1573.70	1	1	1573.70	827
12	89523804	Girik Sangha	smane@example.net	2023-03-24 16:57:27	2023-03-24 16:57:27	994.56	1	1	994.56	826
13	63451391	Amrita Kibe	viswanathanekansh@example.org	2023-03-25 07:26:24	2023-03-25 07:26:24	709.82	1	1	709.82	825
14	29941225	Oviya Bhandari	luke76@example.net	2023-03-26 16:31:03	2023-03-26 16:31:03	145.65	1	1	145.65	824
15	3410758	Qabil Salvi	gandhibhavya@example.net	2023-03-27 11:28:42	2023-03-27 11:28:42	390.03	1	1	390.03	823
16	95736310	Bishakha Vora	sinhasudiksha@example.com	2023-03-27 06:15:38	2023-03-27 06:15:38	688.14	1	1	688.14	823
17	17106161	Gagan Prabhakar	chaitanyaraman@example.net	2023-03-28 08:47:18	2023-03-28 08:47:18	281.73	1	1	281.73	822
18	94137242	Rohan Mallick	wasonmohammed@example.net	2023-03-18 11:29:14	2023-03-28 04:07:24	3062.74	2	1	3062.74	822
19	88684966	Jalsa Khanna	warhi99@example.com	2023-03-30 05:26:58	2023-03-30 05:26:58	595.36	1	1	595.36	820
20	96461062	Udarsh Lal	jhalak01@example.com	2023-03-30 08:43:35	2023-03-30 08:43:35	958.02	1	1	958.02	820
21	64017365	Vasatika Banerjee	warjaschanda@example.net	2023-03-31 08:31:30	2023-03-31 08:31:30	715.72	1	1	715.72	819
22	91602727	Rayaan Palla	yatin72@example.org	2023-04-01 00:44:27	2023-04-01 00:44:27	2906.82	1	1	2906.82	818
23	9423628	Chanakya Dutta	magarekbal@example.net	2023-04-03 14:10:21	2023-04-03 14:10:21	715.72	1	1	715.72	816
24	91228580	Ojasvi Yadav	hiteshsekhon@example.net	2023-04-04 19:47:37	2023-04-04 19:47:37	2127.24	1	1	2127.24	815
25	949062	Jackson Karpe	sembhavika@example.net	2023-04-05 11:28:45	2023-04-05 11:28:45	2840.58	1	1	2840.58	814
26	88779806	Ethan Taneja	mohammed37@example.com	2023-04-06 14:25:38	2023-04-06 14:25:38	882.51	1	1	882.51	813

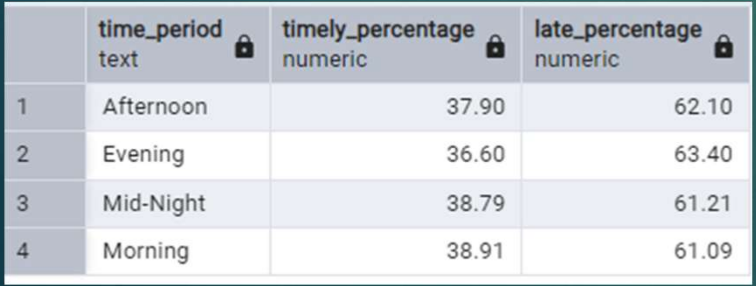
Total rows: 1014   Query complete 00:00:00.180

Delivery Timeliness Analysis

Objective 1: Measure % of on-time vs. late deliveries
Objective 2: See if time of day affects delivery success



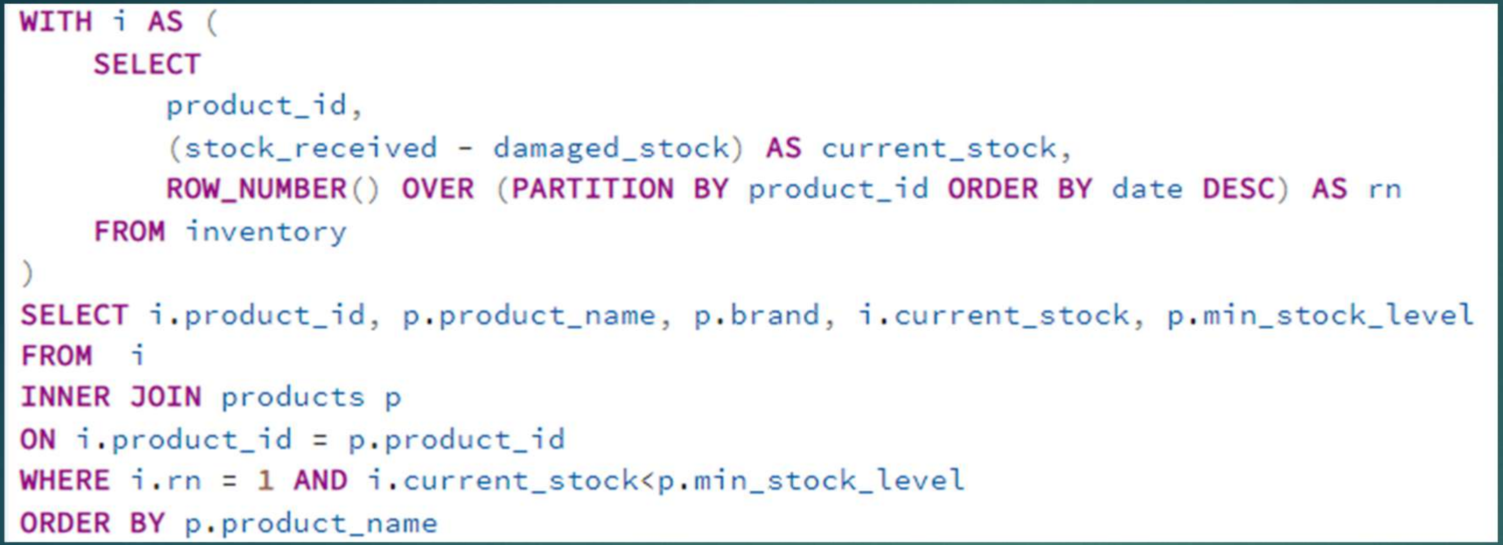
```sql
SELECT
  CASE
    WHEN EXTRACT(HOUR FROM promised_time) BETWEEN 6 AND 11 THEN 'Morning'
    WHEN EXTRACT(HOUR FROM promised_time) BETWEEN 12 AND 17 THEN 'Afternoon'
    WHEN EXTRACT(HOUR FROM promised_time) BETWEEN 18 AND 23 THEN 'Evening'
    ELSE 'Mid-Night'
  END AS time_period,

  ROUND(100.0 * SUM(CASE WHEN delivery_time_minutes <= 0 THEN 1 ELSE 0 END) / COUNT(*), 2) AS timely_percentage,
  ROUND(100.0 * SUM(CASE WHEN delivery_time_minutes > 0 THEN 1 ELSE 0 END) / COUNT(*), 2) AS late_percentage

FROM delivery_performance
GROUP BY time_period
ORDER BY time_period
```



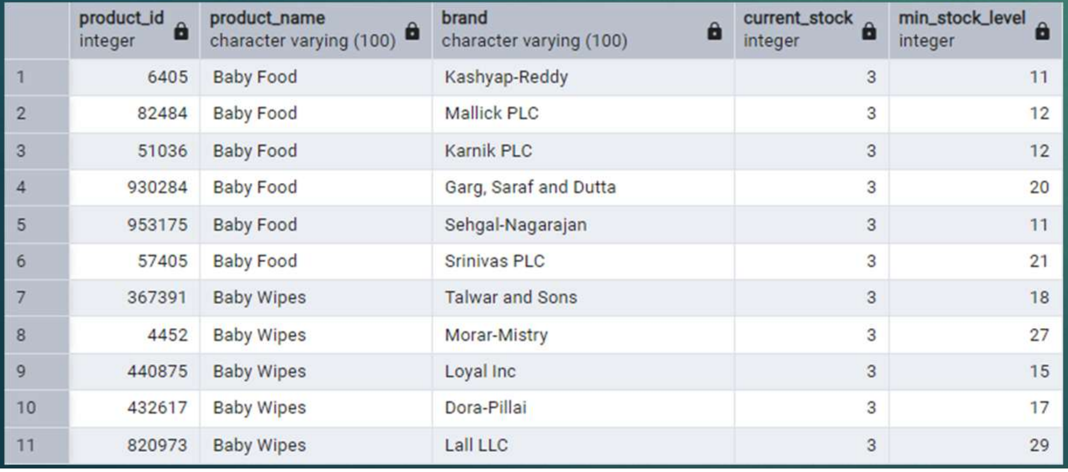
	time_period text	timely_percentage numeric	late_percentage numeric
1	Afternoon	37.90	62.10
2	Evening	36.60	63.40
3	Mid-Night	38.79	61.21
4	Morning	38.91	61.09

Low Stock Product Alert

Identify items where stock is below minimum threshold



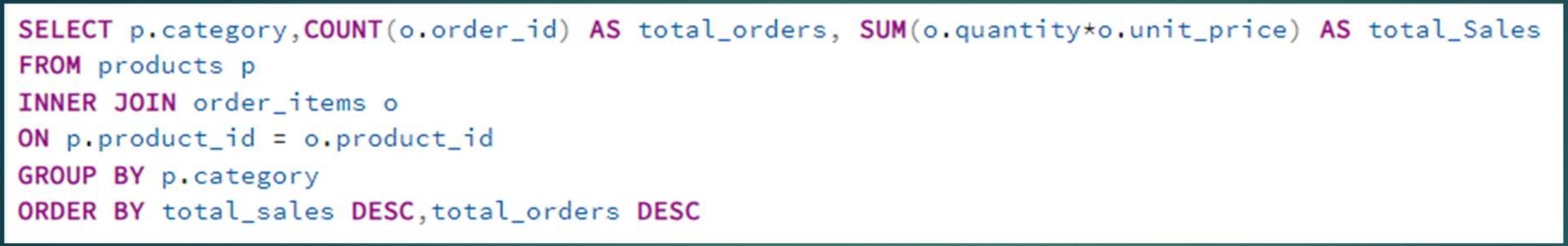
```sql
WITH i AS (
    SELECT
        product_id,
        (stock_received - damaged_stock) AS current_stock,
        ROW_NUMBER() OVER (PARTITION BY product_id ORDER BY date DESC) AS rn
    FROM inventory
)
SELECT i.product_id, p.product_name, p.brand, i.current_stock, p.min_stock_level
FROM  i
INNER JOIN products p
ON i.product_id = p.product_id
WHERE i.rn = 1 AND i.current_stock<p.min_stock_level
ORDER BY p.product_name
```



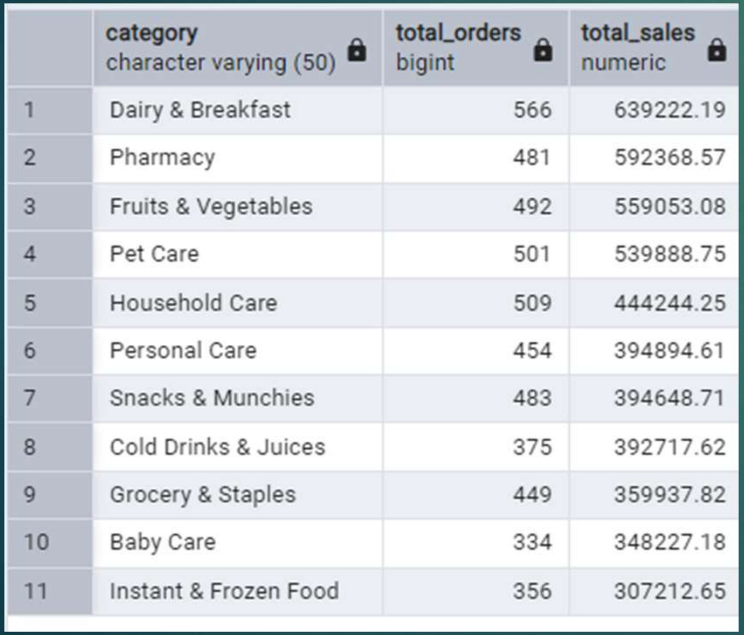
	product_id integer	product_name character varying (100)	brand character varying (100)	current_stock integer	min_stock_level integer
1	6405	Baby Food	Kashyap-Reddy	3	11
2	82484	Baby Food	Mallick PLC	3	12
3	51036	Baby Food	Karnik PLC	3	12
4	930284	Baby Food	Garg, Saraf and Dutta	3	20
5	953175	Baby Food	Sehgal-Nagarajan	3	11
6	57405	Baby Food	Srinivas PLC	3	21
7	367391	Baby Wipes	Talwar and Sons	3	18
8	4452	Baby Wipes	Morar-Mistry	3	27
9	440875	Baby Wipes	Loyal Inc	3	15
10	432617	Baby Wipes	Dora-Pillai	3	17
11	820973	Baby Wipes	Lall LLC	3	29

Sales by Product Category

Track top-selling product categories



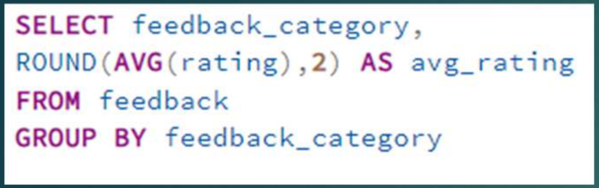
```
SELECT p.category,COUNT(o.order_id) AS total_orders, SUM(o.quantity*o.unit_price) AS total_Sales
FROM products p
INNER JOIN order_items o
ON p.product_id = o.product_id
GROUP BY p.category
ORDER BY total_sales DESC,total_orders DESC
```



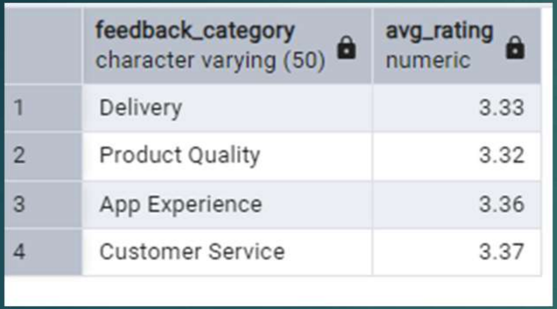
	category character varying (50)	total_orders bigint	total_sales numeric
1	Dairy & Breakfast	566	639222.19
2	Pharmacy	481	592368.57
3	Fruits & Vegetables	492	559053.08
4	Pet Care	501	539888.75
5	Household Care	509	444244.25
6	Personal Care	454	394894.61
7	Snacks & Munchies	483	394648.71
8	Cold Drinks & Juices	375	392717.62
9	Grocery & Staples	449	359937.82
10	Baby Care	334	348227.18
11	Instant & Frozen Food	356	307212.65

Feedback Insights

Understand customer satisfaction trends



```
SELECT feedback_category,
ROUND(AVG(rating),2) AS avg_rating
FROM feedback
GROUP BY feedback_category
```



	feedback_category character varying (50)	avg_rating numeric
1	Delivery	3.33
2	Product Quality	3.32
3	App Experience	3.36
4	Customer Service	3.37

Project Summary & Learnings

Insights Recap:
📊 High-value customers identified
🔁 Churned customers detected for retargeting
🚚 Late deliveries peak in evening
📦 Inventory shortages flagged
💬 Feedback pinpoints service issues
Learnings:
Practical use of SQL window functions and CTEs
Business storytelling using SQL
Query optimization via clean logic and grouping